GABRIELE ROMANATO

Implementare CORS da zero in Node.js con ExpressJS

Il CORS (Cross-Origin Resource Sharing) è un meccanismo che consente di controllare le richieste effettuate da origini diverse rispetto a quella del server. In questo articolo vedremo come implementare manualmente CORS in un'applicazione Node.js con ExpressJS, senza utilizzare middleware esterni come `cors`.

1. Che cos'è CORS?

Quando un'applicazione frontend (come una SPA React) effettua una richiesta HTTP verso un server con un dominio differente, il browser blocca la richiesta a meno che il server non includa specifici header HTTP che ne consentano l'esecuzione. Questi header vengono gestiti tramite la politica CORS.

2. Configurare ExpressJS

Iniziamo creando un'app ExpressJS di base:

```
const express = require('express');
const app = express();
const port = 3000;

app.use(express.json());

app.get('/api/data', (req, res) => {
  res.json({ message: 'Ciao dal server!' });
});

app.listen(port, () => {
  console.log(`Server in ascolto su http://localhost:${port}`);
});
```

3. Gestire CORS manualmente

Per permettere le richieste CORS, dobbiamo intercettare ogni richiesta e impostare correttamente gli header HTTP. Possiamo farlo con un middleware personalizzato.

```
app.use((req, res, next) => {
  // Permetti l'origine specifica o '*'
  res.setHeader('Access-Control-Allow-Origin', '*');

  // Permetti specifici metodi HTTP
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS');
```

```
  // Permetti specifici header
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');

  // Se è una richiesta preflight, termina subito
  if (req.method === 'OPTIONS') {
    return res.sendStatus(204);
  }

  next();
});
```

4. Testare il server

Avvia il server e prova a effettuare una richiesta da un frontend ospitato su un dominio diverso. Grazie agli header impostati, il browser permetterà la comunicazione cross-origin.

5. Miglioramenti possibili

- Limitare `Access-Control-Allow-Origin` solo a origini specifiche
- Gestire dinamicamente l'origine in base alla whitelist
- Rendere configurabile l'elenco di metodi e header consentiti

6. Conclusione

Abbiamo visto come implementare CORS da zero in un'app ExpressJS. Sebbene esistano middleware già pronti come `cors`, comprendere il funzionamento di questi header è fondamentale per scrivere server più sicuri e personalizzabili.

Applicazioni Correlate

Node.js Placeholder Image

Applicazione per la generazione con Node.js di immagini segnaposto.
DockerDocker ComposeNode.jsJavaScriptExpressJS

Node.js URL Shortener

Implementazione in Node.js di un sistema per l'abbreviazione degli URL.

DockerDocker ComposeNode.jsJavaScriptExpressJSMongoDB

- ### JavaScript App Hash Change

  Applicazione che sfrutta gli hash degli URL per gestire contenuto dinamico in JavaScript.
  DockerDocker ComposeNode.jsJavaScriptExpressJSMySQL